

software MBASIC

La gestione dei file

Ecco una nuova rubrica dedicata al software scritto in Basic ... Non è rivolta ad alcun personal in particolare, ma viceversa si interessa di una versione molto usata e nota del Basic, l'MBASIC (il Basic della Microsoft), che i lettori conoscono ormai molto bene e del quale parliamo anche nella rubrica del CP/M.

La scelta di tale versione di Basic non è casuale: i programmi presentati potranno girare senza alcuna modifica in tutti i personal dotati di MBASIC, quindi ad esempio tutti quelli in CP/M. Ci siamo rigorosamente imposti di non far comparire alcuna PE-EK o POKE, strettamente legata alla macchina e perciò ci occuperemo di nozioni e programmi generali.

Starà poi all'abilità del lettore aggiungere tutti quegli abbellimenti formali (menu, output su stampante, ecc.), in prima analisi non necessari.

I file e la loro gestione

Per "file" si intende, e lo diciamo per chi non lo sapesse, un insieme di dati (valori numerici e/o stringhe) organizzati secondo certe regole e generalmente memorizzati su disco: diciamo generalmente perché volutamente trascureremo la gestione dei file in un sistema dotato di registratore a cassetta, a causa degli enormi problemi che un tale supporto di memoria di massa comporta.

Parlando di insieme di dati, diciamo anche che deve essere dotato di una seppur minima struttura, sia a livello di singoli dati che per gruppi di essi.

Un esempio di file è classico: la rubrica telefonica, in genere il primo tipo di file su cui un programmatore si cimenta.

In una rubrica telefonica abbiamo una struttura nonché una certa omogeneità dei dati: la struttura consiste nel fatto che la singola informazione (il dato, detto "record") è costituita da più parti (nome, cognome, indirizzo, numero di telefono), le quali a loro volta sono omogenee: tutti i nomi, e tutti i cognomi sono esprimibili

con stringhe, tutti gli indirizzi con stringhe alfanumeriche e i numeri di telefono sono ... numeri.

Per quanto riguarda questi ultimi, in genere conviene pensarli anche loro codificati con stringhe, dal momento che in tal modo si possono inserire i prefissi separati dal numero da una barra ("/"), altrimenti intraducibile.

Parlato di file in genere, aggiungiamo che esistono due differenti modalità di gestione, a loro volta strettamente legate alla struttura stessa del file.

Sequenziali o Random?

Cominciamo dai primi. Si dicono file sequenziali in quanto tale è l'accesso ai

singoli dati che lo compongono, dal momento che i dati vengono memorizzati sul dischetto uno di seguito all'altro.

Vediamo cosa significa tutto questo: può sembrare infatti strano visto che usiamo dischetti e non cassette magnetiche.

Anche se dobbiamo scrivere i dati uno dopo l'altro, più o meno "separati" l'uno dal successivo, anche se per leggere un certo dato siamo costretti a "scorrere" tutto il file, dato per dato, anche se per aggiungere nuovi dati ad un file dobbiamo ricopiarlo, fare le aggiunte e scriverlo di nuovo, malgrado tutto ciò, dicevamo, i file sequenziali "resistono" ancora grazie alla loro estrema facilità d'uso, generalità, elasticità.

Ma vediamo di confrontarli con i "random": in questo caso i dati possono essere

FILES SEQUENZIALI (Scrittura)	
Funzione logica	Istruzione dell'MBASIC
1 - Apertura del file in output	OPEN "0", [#] buffer, "nomefile"
2 - Scrittura dei dati	PRINT# buffer, [USING stringa ;] variabili WRITE# buffer, variabili
3 - Chiusura del file	CLOSE [#] [buffer]
Tabella 1	
FILES SEQUENZIALI (Lettura)	
Funzione logica	Istruzione dell'MBASIC
1 - Apertura del file in input	OPEN "I", [#] buffer, "nomefile"
2 - Lettura dei dati	INPUT# buffer, variabili LINE INPUT# buffer, variabili
3 - Chiusura del file	CLOSE [#] [buffer]
Tabella 2	

I file sequenziali prevedono due modalità di accesso in scrittura ed in lettura. Si avranno in entrambi i casi tre operazioni logiche da compiere a cui corrispondono varie istruzioni MBASIC.

letti, scritti e aggiunti a "caso" (random appunto) e cioè in qualunque posizione, senza dover scomodare i rimanenti. Sembrerebbero perciò ideali ...

In realtà invece è proprio la gestione dal punto di vista programmatico ad essere veramente pesante e non del tutto intuitiva: ciò che si guadagna in velocità nell'accedere direttamente ad un certo record, piano piano si perde tra conversioni e posizionamenti di stringhe, operazioni queste molto "care" in termini di tempo di esecuzione.

Comunque, alla fine della prossima puntata, quando avremo conosciuto bene anche i file random, potremo giudicare quale dei due servirà meglio al nostro scopo.

I file sequenziali

Torniamo alla nostra rubrica telefonica: decidiamo perciò, come detto, di codificare cognomi, nomi e indirizzi con stringhe alfanumeriche ed i numeri telefonici con numeri, proprio per vedere come solo con i sequenziali si possono trattare i numeri puri.

Utilizzeremo rispettivamente le stringhe C\$, N\$, I\$ e la variabile numerica N: per le stringhe non abbiamo alcuna limitazione di lunghezza (a parte la naturale limitazione a 255 caratteri), cosa che nei "random" avrà una notevolissima importanza, mentre supporremo, ma è solo un esempio, che i numeri di telefono arrivino al massimo a sei cifre, per non incorrere in brutti passaggi a notazione esponenziale.

Per quanto riguarda la lunghezza dei cognomi, nomi e indirizzi possiamo tranquillamente spaziare tra (ogni riferimento è puramente casuale!!) "Ugo" e "Giambartolomeo", "Re" e "Piccolomini", abitanti a "Via Po" e alla "Circonvallazione Gianicolense". Definite perciò queste quantità, avremo un record di lunghezza variabile entro vasti limiti.

Stabiliamo ora di dare un nome al nostro file, così come siamo abituati a fare con i nostri programmi all'atto del salvataggio su disco: anche i programmi sono dei file!!

L'ultima informazione di cui abbiamo bisogno nel gestire questo file è la modalità di accesso: in lettura o in scrittura.

A seconda della scelta fatta dobbiamo seguire un certo schema logico, in cui ad ogni operazione logica corrispondono una o più funzioni MBASIC: teniamo perciò sott'occhio le tabelle 1 e 2 relative ai due casi prospettati.

In particolare in tali tabelle abbiamo indicato con maiuscolo le istruzioni MBASIC ed in minuscolo i valori che dobbiamo inserire, mentre con le parentesi quadre abbiamo indicato ciò che può essere messo o meno a seconda dei gusti e delle necessità. Come esempi dei due modi di accesso, consideriamo i due programmi n° 1 e n° 2, relativi ad un semplicissimo file "agenda" e

ai dati di cui abbiamo parlato sopra. Confrontando i due schemi logici, vediamo che in fondo sono molto simili: c'è una prima fase di apertura del file in cui si specificano alcune caratteristiche, c'è una seconda fase di accesso (in lettura o in scrittura) ai dati ed una terza fase di chiusura del file.

Proprio per questo procederemo parallelamente nell'analisi.

Per quanto riguarda l'apertura o attivazione del file, troviamo l'istruzione OPEN seguita da "i" se vogliamo leggere e da "o" se vogliamo scrivere i dati; successivamente dobbiamo mettere un numero, precedente o meno dal simbolo #, indicante un certo

due tipi di "separatori" rappresentati dalle virgole (,) o dalle virgolette ("), che devono in entrambi i casi essere "forzati" dal programmatore.

Ma mentre per le virgole è sufficiente indicarle come stringhe esplicite (","), per le virgolette ciò non vale (dato che una stringa del tipo "" non sarebbe interpretabile facilmente!) ed allora si è costretti ad usare l'espressione CHR\$(34).

Per snellire ancora il tutto suggeriamo

```

5 REM      SCRITTURA DI UN FILE SEQUENZIALE
10 OPEN"o",1,"agenda":REM      apertura in output
20 INPUT"cognome";C$:REM      input dei dati
30 INPUT"nome";N$
40 INPUT"indirizzo";I$
50 INPUT"numero di telefono";N
60 PRINT#1,C$;" ";N$;" ";I$;" ";N:REM      scrittura su disco
70 INPUT"ancora";R$
80 IF R$ = "N" THEN CLOSE:END:REM      chiusura del file
90 C$=" ";N$=" ";I$=" ";N:REM      azzeramento stringhe
100 GOTO20:REM      rientro nel loop
    
```

Programma 1

```

5 REM      LETTURA DI UN FILE SEQUENZIALE
10 CLEAR200:REM      spazio per le stringhe
20 OPEN"i",1,"agenda":REM      apertura in input
30 IF EOF(1) THEN CLOSE:END:REM      chiusura del file
40 INPUT#1,C$,N$,I$,N:REM      lettura dal disco
50 PRINTC$;" ";N$;" ";I$;" ";N:REM      output su video
60 GOTO40:REM      rientro nel loop
    
```

Programma 2

buffer di memoria che utilizziamo per la memorizzazione dei dati in transito.

Tale valore potrà essere compreso tra 1 e 15 e rimarrà associato a tale buffer ed al file stesso fino a che non chiuderemo il file in questione.

Dopo aver perciò "svegliato" il nostro file, possiamo effettuare l'operazione di accesso ai dati: consideriamo dapprima la scrittura per poi arrivare alla lettura.

Per registrare dati sul dischetto abbiamo a disposizione l'istruzione PRINT#, che però ci costringe ad usare un artificio per separare fisicamente i dati che costituiscono il record.

Infatti la sintassi dell'istruzione richiede che le variabili ed in genere le stringhe siano separate (nel comando!) da un punto e virgola (";"): ciò può comportare alcuni piccoli problemi, di facile risoluzione se pensiamo di usare una normale PRINT sul video anziché su disco.

Se noi scriviamo infatti PRINT A\$; B\$; C\$ dove A\$, B\$ e C\$ sono rispettivamente pari a "MILANO", "TORINO" e "ROMA", sul video otterremo la scritta MILANOTORINOROMA e cioè i tre nomi tutti attaccati. Esattamente accade per l'istruzione PRINT#.

Allora per separare i vari dati si usano

ad esempio di usare variabili tipo:

V\$ = "," e W\$ = CHR\$(34)

Nel nostro caso l'istruzione corretta, usando le "virgole" sarà:

PRINT#1, A\$;" ";B\$;" ";C\$

oppure

PRINT#1, A\$;V\$;B\$;V\$;C\$

Usando invece le virgolette dobbiamo scrivere qualcosa come:

PRINT#1, CHR\$(34);A\$;CHR\$(34);CHR\$(34);B\$;CHR\$(34) ecc.

oppure

PRINT#1, W\$;A\$;W\$;W\$;B\$;W\$;W\$;C\$;W\$

In fin dei conti, se usiamo la virgola come separatore fisico tra i vari campi, otteniamo nel primo caso, su dischetto:

MILANO,TORINO,ROMA

Nel caso delle virgolette, dobbiamo scriverle prima e dopo ogni dato del record (come una sorta di parentesi), avendosi in tal modo sul dischetto:

"MILANO" "TORINO" "ROMA"

Per capire meglio questo marchingegno, consideriamo ora come vengono trattati i numeri.

Nel caso del video un'istruzione del tipo PRINT A;B;C comporta, a differenza delle stringhe, che i numeri rimangono lo stesso separati in quanto per loro natura (e convenzione!) hanno sicuramente uno spa-

zio bianco "in coda", mentre "in testa" hanno il segno che può essere "-" oppure un altro spazio bianco.

Perciò, se i numeri sono positivi li vedremo senz'altro ben separati: anche nel caso di valori negativi almeno uno spazio bianco rimarrà.

Inutile dire che tutto questo si rispecchia nell'output su dischetto: rimane soltanto da aggiungere che il numero, cifra per cifra, è codificato in ASCII e perciò poca è la differenza di "trattamento" tra quantità intere, reali o in doppia precisione: tanto è lungo il numero sul video, tanto occuperà sul disco.

Sarà ora facile capire cosa può succedere se mischiamo tra loro quantità numeriche con stringhe: in alcuni casi i corrispondenti caratteri su disco saranno separati ed in altri saranno contigui. Per rimediare a ciò si possono usare indistintamente (ma a ragione veduta!) sia le virgole che le virgolette.

Certo questo non poteva andare bene, in quanto troppo macchinoso: ecco che perciò gli ideatori dell'MBASIC si sono inventati l'istruzione WRITE#, che, con un semplice colpo di spugna, cancella tutte queste cavillosità.

Volendo inframmezzare stringhe a dati numerici, infatti, conviene scrivere ad esempio

```
WRITE#1, A$,A,B,G$,I$,N,"PIPP0"
```

Non dimentichiamoci infatti che oltre

alle variabili stringa, possiamo anche scrivere direttamente delle stringhe (vedi "pip-po"). Con l'istruzione WRITE# abbiamo che tutte le stringhe sono precedute e seguite dalle virgolette, mentre tutti i dati sono indistintamente separati da una virgola.

Abbiamo dimenticato di dire che in tutti i casi visti finora, alla fine di ogni record viene inserito automaticamente un "carriage return" (CR o CHR\$(13)). Almeno in questo modo i record sono riconoscibili!

Tutto questo discorso, se vogliamo quanto contorto (ma la colpa è delle istruzioni!), perché poi, in fin dei conti, quei benedetti dati che scriviamo sul disco li vogliamo andare a rileggere!

Si può ben immaginare che cosa potrebbe succedere se invece di tre stringhe (come nel caso di MILANO, TORINO e ROMA) l'istruzione di INPUT# trovasse una "stringona" data da MILANOTORINO-ROMA. Le altre due stringhe dove le va a prendere? Magari nel dato successivo, originariamente numerico!?

Ecco che, usando come separatori le virgole o le virgolette oppure l'istruzione WRITE#, facciamo un notevole favore all'INPUT#! Se invece vogliamo leggere "tutto" il record, indipendentemente dal fatto che contenga eventuali numeri, possiamo usare l'istruzione LINEINPUT#. Tra l'altro questa ci consente di leggere eventuali virgole o virgolette "effettiva-

mente" facenti parte dei singoli dati. Se ci vogliamo complicare ulteriormente la vita, infatti, immaginiamo di voler salvare su disco un record in cui un dato, per chissà quale strano motivo è il seguente:

```
SI ALZÒ, DICENDO: "ARRIVEDERCI"
```

Come si vede, continue virgole; virgolette, blank, due punti e chi più ne ha più ne metta.

Trovato il modo di memorizzare tale stringa grazie all'uso inopinato di CHR\$ e/o di variabili-stringa ausiliarie, con una semplice LINEINPUT# andremo a leggere il tutto.

Tornando alle istruzioni di output, abbiamo, nel caso di scrittura di numeri, la possibilità di formattarli con l'USING, con le identiche regole applicabili alla PRINTUSING: ne riparleremo comunque in seguito.

Terminiamo questa puntata con un'ultima dolente nota.

Aggiunta di record ad un file sequenziale

Una volta creato il file "agenda" con il programma n° 1 e verificati i dati introdotti, grazie al programma n° 2, se vogliamo aggiungere ulteriori record NON dobbiamo assolutamente usare il primo programma: in questo caso, infatti, otterremo la cancellazione dei dati preesistenti ed il

2500 PROGRAMMI PER SPECTRUM E COMMODORE

COMPUTER HOUSE

SOFTWARE: il più vasto assortimento di programmi per COMMODORE, SPECTRUM e AMSTRAD, dai giochi più entusiasmanti ai gestionali più potenti. Decine di programmi in arrivo ogni settimana direttamente dagli Stati Uniti e dall'Inghilterra. Tutto il software a prezzi bassissimi; cassette originali a partire da £. 7.000 !!

HARDWARE: i migliori home computers del mercato internazionale corredati da tutte le periferiche esistenti e dall'assistenza di esperti programmatori; e naturalmente i prezzi migliori!

In **ESCLUSIVA** per l'Italia programma di hardcopy in linguaggio macchina che rende grafiche le stampanti MPS 802 e 1526. Permette di stampare qualunque tipo di immagine creata con il Koala, Doodle e tutto il software grafico in commercio.

Dischetti magnetici a partire da £. 3.500

Per qualunque problema richiedete gratuitamente la nostra consulenza ed il nostro completissimo catalogo di 40 pagine a:

COMPUTER HOUSE di Somenzi Dino
Via S. Francesco 15, 41012 Carpi (MO), Tel. 059 / 693528
Via Secchi 28/B, 42100 Reggio Emilia

PREZZI STRAORDINARI.

DISPONIBILI SUBITO I NUOVISSIMI: COMMODORE + 4 E C 16, L'ECONOMICO AMSTRAD E IL POTENTISSIMO QL

Prezzi particolari rivenditori.

```

5 REM      AGGIUNTA RECORD AD UN FILE SEQUENZIALE
10 OPEN"i",1,"agenda":REM
20 OPEN"o",2,"backup":REM
30 IF EOF(1) THEN 70:REM
40 LINEINPUT#1,A$:REM
50 PRINT#2,A$:REM
60 GOTO30:REM
70 CLOSE1:REM
80 INPUT"cognome";C$:REM
90 INPUT"nome";N$
100 INPUT"indirizzo";I$
110 INPUT"numero di telefono";N
120 PRINT#2,C$;",";N$;",";I$;",";N:REM
130 INPUT"ancora";R$
140 IF R$ = "s" THEN C$="":N$="":I$="":GOTO80:REM
150 CLOSE:REM
160 KILL"agenda":REM
170 NAME "backup" AS "agenda":REM
    
```

apertura in input
 apertura in output
 salto se end-of-file
 copia di "agenda"
 su "backup"
 loop di copia
 chiuso file "agenda"
 input nuovi dati

 scrittura su disco

 rientro nel loop
 chiuso anche "backup"
 cancellazione di "agenda"
 creazione del nuovo file

Programma 3

nuovo file ottenuto conterrà **SOLTANTO** i record aggiunti!

Per operare correttamente dobbiamo perciò usare un programma adatto (il n° 3) che, in poche parole, effettua le seguenti operazioni:

- ricopia pari pari il file "agenda" sul file "backup", il quale (badate bene! è importante!) è stato aperto "in output";
- immette i nuovi record nel file "backup", in quanto "agenda" era stato aperto in "input" e su di esso non si può scrivere;
- cancella l'ormai vecchio ed inutile "agenda";

— rinomina il nuovo file "backup" con il nome "agenda".

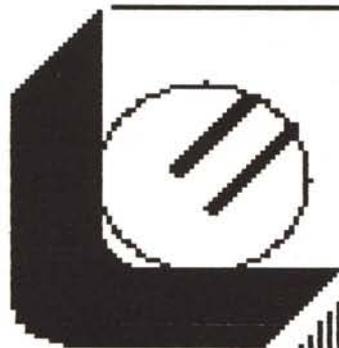
Importante è notare che un file non si può aprire contemporaneamente in lettura e in scrittura, e perciò dobbiamo ricorrere ad un file ausiliario dove andare poi a scrivere i nuovi record. Ciò può comportare gravissimi problemi di spazio, specie se il file di partenza è grande.

Da notare, nel programma n° 3, l'uso della LINEINPUT# per la lettura completa ed indiscriminata dei singoli caratteri, fino ad incontrare un "carriage return" oppure fino ad un massimo di 255 caratteri.

Chiudiamo dunque il discorso sui "sequenziali" segnalando due ulteriori funzioni di supporto:

- EOF (buffer) è una funzione booleana che segnala se si è arrivati o no alla fine del file (si vedano le linee 30 dei programmi n° 2 e 3);
- LOC (buffer) indica quanti blocchi da 128 byte sono stati letti o scritti da quando abbiamo aperto un file con una OPEN.

Arriveremo alla prossima puntata in cui analizzeremo i pro e i contro dei file "random" e della loro gestione. **MC**



SETTORE INFORMATICA
BY

L & E ENGINEERING

via c.salentina 21 - 73045 Ieverano

IL NOSTRO COMPUTER COMPATIBILE CON TUTTI I FRUTTI..VI SORPRENDERA 'ANCORA DI PIU'.

> per voi l'hardware piu' sofisticato... e tanto software gia' disponibile.

L'ASSISTENZA TECNICA NASCE DA OLTRE DIECI ANNI DI PROGETTAZIONI ELETTRONICHE

tel.0832/925039 tix.860219 libele i